Concurrent separation logic and operational semantics

Viktor Vafeiadis

Max Planck Institute for Software Systems (MPI-SWS)

Paper summary

- Yet another soundness proof for CSL
- Simple
  - Based on a standard operational semantics
  - Novel semantic definition of triples fits on a slide
- Extensible
  - Permissions
  - RGSep
  - Storable locks [Buisse, Birkedal, Støvring, MFPS 2011]
  - Concurrent abstract predicates [ECOOP 2010]
- Explains precision & conjunction rule
- Fully mechanized in Isabelle/HOL

The meaning of CSL judgments

$$\Gamma \models \{P\}C\{Q\} \text{ iff } \forall s\, h\, n.\ s, h \models P \implies \mathrm{safe}_n(C, s, h, \Gamma, Q)$$

$$\mathrm{safe}_0(C, s, h, \Gamma, Q) \stackrel{\text{def}}{=} \mathit{true}$$

$$\begin{aligned}
&\mathrm{safe}_{n+1}(C, s, h, \Gamma, Q) \stackrel{\text{def}}{=} \\
&\quad (C = \textbf{skip} \implies s, h \models Q) \\
&\quad \wedge (\forall h_{\mathrm{F}}.\ \neg(C, s, h \uplus h_{\mathrm{F}}) \to \textbf{abort}) \\
&\quad \wedge (\forall h_{\mathrm{J}}\, h_{\mathrm{F}}\, C'\, s'\, h'.\ (C, s \uplus h_{\mathrm{J}} \uplus h_{\mathrm{F}}, h) \to (C', s', h') \\
&\qquad\qquad \wedge s, h_{\mathrm{J}} \models \circledast_{r \in \mathit{Locked}(C') \setminus \mathit{Locked}(C)} \Gamma(r) \\
&\qquad \implies \exists h''\, h'_{\mathrm{J}}.\ h' = h'' \uplus h'_{\mathrm{J}} \uplus h_{\mathrm{F}} \\
&\qquad\qquad \wedge s, h'_{\mathrm{J}} \models \circledast_{r \in \mathit{Locked}(C) \setminus \mathit{Locked}(C')} \Gamma(r) \\
&\qquad\qquad \wedge \mathrm{safe}_n(C', s', h'', \Gamma, Q))
\end{aligned}$$

That's all!

Sequential language

$$
\begin{aligned}
E &::= x \mid n \mid E + E \mid E - E \mid \ldots \\
B &::= B \wedge B \mid \neg B \mid E = E \mid E \leq E \mid \ldots \\
C &::= \textbf{skip} \mid x := E \mid x := [E] \mid [E] := E \mid x := \textbf{alloc}(E) \mid \textbf{free}(E) \\
&\quad \mid C_1; C_2 \mid \textbf{if } B \textbf{ then } C_1 \textbf{ else } C_2 \mid \textbf{while } B \textbf{ do } C
\end{aligned}
$$

- Small-step operational semantics:

$$
(C, s, h) \to (C', s', h') \qquad (C, s, h) \to \textbf{abort}
$$

$$
s : \mathsf{Stack} \stackrel{\text{def}}{=} \mathsf{VarName} \to \mathsf{Val}
$$

$$
h : \mathsf{Heap} \stackrel{\text{def}}{=} \mathsf{Loc} \rightharpoonup \mathsf{Val}
$$

- Rules for sequential composition:

$$
(\textbf{skip}; C, s, h) \to (C, s, h) \qquad \frac{(C_1, s, h) \to (C_1', s', h')}{(C_1; C_2, s, h) \to (C_1'; C_2, s', h')}
$$

Parallel composition

$$C ::= \ldots \mid C_1 \| C_2$$

- Interleaving semantics:

$$\frac{(C_1, s, h) \to (C_1', s', h')}{(C_1 \| C_2, s, h) \to (C_1' \| C_2, s', h')} \qquad \frac{(C_2, s, h) \to (C_2', s', h')}{(C_1 \| C_2, s, h) \to (C_1 \| C_2', s', h')}$$

- Abort semantics:

$$\frac{(C_1, s, h) \to \textbf{abort}}{(C_1 \| C_2, s, h) \to \textbf{abort}} \qquad \frac{(C_2, s, h) \to \textbf{abort}}{(C_1 \| C_2, s, h) \to \textbf{abort}}$$

- Termination:

$$(\textbf{skip} \| \textbf{skip}, s, h) \to (\textbf{skip}, s, h)$$

Atomic blocks

$$C ::= \ldots \mid \textbf{atomic } C$$

- Atomic blocks execute in one step

$$\frac{(C, s, h) \rightarrow^* (\textbf{skip}, s', h')}{(\textbf{atomic } C, s, h) \rightarrow (\textbf{skip}, s', h')}$$

$$\frac{(C, s, h) \rightarrow^* \textbf{abort}}{(\textbf{atomic } C, s, h) \rightarrow \textbf{abort}}$$

- Normally, also need a rule such as

$$\frac{(C, s, h) \rightarrow^\omega}{(\textbf{atomic } C, s, h) \rightarrow (\textbf{atomic } C, s, h)}$$

for atomic blocks that don't terminate

Multiple resources

- Lock declarations & conditional critial regions (CCRs)

$$C ::= \ldots \mid \textbf{resource } r \textbf{ in } C \mid \textbf{with } r \textbf{ when } B \textbf{ do } C \mid \textbf{within } r \textbf{ do } C$$

- Enter a CCR

$$\frac{[\![B]\!](s)}{(\textbf{with } r \textbf{ when } B \textbf{ do } C, s, h) \rightarrow (\textbf{within } r \textbf{ do } C, s, h)}$$

- Execute body of a CCR

$$\frac{(C, s, h) \rightarrow (C', s', h') \qquad r \notin Locked(C')}{(\textbf{within } r \textbf{ do } C, s, h) \rightarrow (\textbf{within } r \textbf{ do } C', s', h')}$$

- Exit the CCR

$$(\textbf{within } r \textbf{ do skip}, s, h) \rightarrow (\textbf{skip}, s, h)$$

Rules for parallel composition

$$\frac{(C_1, s, h) \to (C_1', s', h') \quad Locked(C_1') \cap Locked(C_2) = \emptyset}{(C_1 \| C_2, s, h) \to (C_1' \| C_2, s', h')} \qquad \frac{(C_2, s, h) \to (C_2', s', h') \quad Locked(C_1) \cap Locked(C_2') = \emptyset}{(C_1 \| C_2, s, h) \to (C_1 \| C_2', s', h')}$$

where

$$Locked(C) \stackrel{\text{def}}{=} \{r \mid \exists C'.\ (\textbf{within } r \textbf{ do } C') \text{ is a subterm of } C\}$$

Ensures that commands are well-formed:

$$wf(\textbf{skip}) \stackrel{\text{def}}{=} true$$
$$wf(C_1; C_2) \stackrel{\text{def}}{=} wf(C_1) \wedge wf(C_2) \wedge (Locked(C_2) = \emptyset)$$
$$wf(C_1 \| C_2) \stackrel{\text{def}}{=} wf(C_1) \wedge wf(C_2) \wedge (Locked(C_1) \cap Locked(C_2) = \emptyset)$$
$$wf(\textbf{with } r \textbf{ when } B \textbf{ do } C) \stackrel{\text{def}}{=} wf(C) \wedge (Locked(C) = \emptyset)$$
$$wf(\textbf{within } r \textbf{ do } C) \stackrel{\text{def}}{=} wf(C) \wedge r \notin Locked(C)$$

Some proof rules

$$\frac{\Gamma \vdash \{P_1\}C_1\{Q_1\} \qquad fv(\Gamma, P_1, C_1, Q_1) \cap wr(C_2) = \emptyset \quad \Gamma \vdash \{P_2\}C_2\{Q_2\} \qquad fv(\Gamma, P_2, C_2, Q_2) \cap wr(C_1) = \emptyset}{\Gamma \vdash \{P_1 * P_2\}C_1 \| C_2\{Q_1 * Q_2\}} \quad (\textsc{Par})$$

$$\frac{\Gamma \vdash \{(P * J) \wedge B\}C\{Q * J\}}{\Gamma, r : J \vdash \{P\}\textbf{with } r \textbf{ when } B \textbf{ do } C\{Q\}} \quad (\textsc{With})$$

$$\frac{\Gamma, r : J \vdash \{P\}C\{Q\} \qquad fv(J) \cap wr(C) = \emptyset}{\Gamma \vdash \{P * J\}\textbf{resource } r \textbf{ in } C\{Q * J\}} \quad (\textsc{Res})$$

$$\frac{\Gamma \vdash \{P\}C\{Q\} \qquad fv(R) \cap wr(C) = \emptyset}{\Gamma \vdash \{P * R\}C\{Q * R\}} \quad (\textsc{Frame})$$

- NB: Draconian variable side-conditions.

Proof rules for atomic blocks

$$\frac{\begin{array}{ll} J \vdash \{P_1\}C_1\{Q_1\} & fv(J, P_1, C_1, Q_1) \cap wr(C_2) = \emptyset \\ J \vdash \{P_2\}C_2\{Q_2\} & fv(J, P_2, C_2, Q_2) \cap wr(C_1) = \emptyset \end{array}}{J \vdash \{P_1 * P_2\}C_1 \| C_2\{Q_1 * Q_2\}} \quad (\textsc{Par})$$

$$\frac{\mathbf{emp} \vdash \{P * J\}C\{Q * J\}}{J \vdash \{P\}\mathbf{atomic}\ C\{Q\}} \quad (\textsc{Atom})$$

$$\frac{J * R \vdash \{P\}C\{Q\} \qquad fv(R) \cap wr(C) = \emptyset}{J \vdash \{P * R\}C\{Q * R\}} \quad (\textsc{Share})$$

$$\frac{J \vdash \{P\}C\{Q\} \qquad fv(R) \cap wr(C) = \emptyset}{J \vdash \{P * R\}C\{Q * R\}} \quad (\textsc{Frame})$$

Hoare triples (partial correctness)

$$\models \{P\}C\{Q\}$$

if and only if

$$\forall s\, h\, s'\, h'.\ s, h \models P \ \wedge\ (C, s, h) \rightarrow^* (\mathbf{skip}, s', h') \implies s', h' \models Q$$

if and only if

$$\forall s\, h.\ s, h \models P \implies (\forall s'\, h'.\ (C, s, h) \rightarrow^* (\mathbf{skip}, s', h') \implies s', h' \models Q)$$

if and only if

$$\forall s\, h.\ s, h \models P \implies (\forall m.\ \forall s'\, h'.\ (C, s, h) \rightarrow^m (\mathbf{skip}, s', h') \implies s', h' \models Q)$$

if and only if

$$\forall s\, h\, n.\ s, h \models P \implies (\forall m < n.\ \forall s'\, h'.\ (C, s, h) \rightarrow^m (\mathbf{skip}, s', h') \implies s', h' \models Q)$$

Configuration safety

$$\models \{P\}C\{Q\} \text{ iff } \forall s\, h\, n.\ s, h \models P \implies \mathrm{safe}_n(C, s, h, Q)$$

where

$$\mathrm{safe}_n(C, s, h, Q) \stackrel{\text{def}}{=} \\ (\forall m < n.\ \forall s'\, h'.\ (C, s, h) \rightarrow^m (\textbf{skip}, s', h') \implies s', h' \models Q)$$

As an inductive definition:

$$\mathrm{safe}_0(C, s, h, Q) = \mathit{true}$$
$$\mathrm{safe}_{n+1}(C, s, h, Q) = \\ (C = \textbf{skip} \implies s, h \models Q) \\ \wedge (\forall C'\, s'\, h'.\ (C, s, h) \rightarrow (C', s', h') \\ \implies \mathrm{safe}_n(C', s', h', Q))$$

Configuration safety

$$\models \{P\}C\{Q\} \text{ iff } \forall s\, h\, n.\ s, h \models P \implies \mathrm{safe}_n(C, s, h, Q)$$

$$\mathrm{safe}_0(C, s, h, Q) \stackrel{\text{def}}{=} \mathit{true}$$

$$\begin{aligned}
&\mathrm{safe}_{n+1}(C, s, h, Q) \stackrel{\text{def}}{=} \\
&\quad (C = \textbf{skip} \implies s, h \models Q) \\
&\quad \wedge (\forall C'\, s'\, h'.\ (C, s, h) \to (C', s', h') \\
&\qquad \implies \mathrm{safe}_n(C', s', h', Q))
\end{aligned}$$

Fault-avoidance

$$\models \{P\}C\{Q\} \text{ iff } \forall s\, h\, n.\ s, h \models P \implies \mathrm{safe}_n(C, s, h, Q)$$

$$\mathrm{safe}_0(C, s, h, Q) \stackrel{\text{def}}{=} \mathit{true}$$

$$\begin{aligned}
&\mathrm{safe}_{n+1}(C, s, h, Q) \stackrel{\text{def}}{=} \\
&\quad (C = \textbf{skip} \implies s, h \models Q) \\
&\quad \wedge (\neg(C, s, h) \to \textbf{abort}) \\
&\quad \wedge (\forall C'\, s'\, h'.\ (C, s, h) \to (C', s', h') \\
&\qquad \implies \mathrm{safe}_n(C', s', h', Q))
\end{aligned}$$

- "Well-specified programs don't go wrong."

"Bake in" the frame rule

$$\models \{P\}C\{Q\} \text{ iff } \forall s\, h\, n.\ s, h \models P \implies \mathrm{safe}_n(C, s, h, Q)$$

$$\mathrm{safe}_0(C, s, h, Q) \stackrel{\text{def}}{=} \mathit{true}$$

$$\begin{aligned}
&\mathrm{safe}_{n+1}(C, s, h, Q) \stackrel{\text{def}}{=} \\
&\quad (C = \textbf{skip} \implies s, h \models Q) \\
&\wedge (\forall h_{\mathrm{F}}.\ \neg(C, s, h \uplus h_{\mathrm{F}}) \rightarrow \textbf{abort}) \\
&\wedge (\forall h_{\mathrm{F}}\ C'\ s'\ h'.\ (C, s \uplus h_{\mathrm{F}}, h) \rightarrow (C', s', h') \\
&\qquad \implies \exists h''.\ h' = h'' \uplus h_{\mathrm{F}} \wedge \mathrm{safe}_n(C', s', h'', Q))
\end{aligned}$$

- No safety monotonicity & frame property
- Same definition works for permissions ($\uplus \rightsquigarrow$ addition of permission-heaps)

Atomic blocks

$$J \models \{P\}C\{Q\} \text{ iff } \forall s\, h\, n.\ s, h \models P \implies \mathrm{safe}_n(C, s, h, J, Q)$$

$$\mathrm{safe}_0(C, s, h, J, Q) \stackrel{\text{def}}{=} \mathit{true}$$

$$\begin{aligned}
&\mathrm{safe}_{n+1}(C, s, h, J, Q) \stackrel{\text{def}}{=} \\
&\quad (C = \textbf{skip} \implies s, h \models Q) \\
&\quad \wedge (\forall h_{\mathrm{J}}\, h_{\mathrm{F}}.\ s, h_{\mathrm{J}} \models J \implies \neg(C, s, h \uplus h_{\mathrm{J}} \uplus h_{\mathrm{F}}) \to \textbf{abort}) \\
&\quad \wedge (\forall h_{\mathrm{J}}\, h_{\mathrm{F}}\, C'\, s'\, h'.\ (C, s \uplus h_{\mathrm{J}} \uplus h_{\mathrm{F}}, h) \to (C', s', h') \\
&\qquad\qquad \wedge s, h_{\mathrm{J}} \models J \\
&\qquad \implies \exists h''\, h'_{\mathrm{J}}.\ h' = h'' \uplus h'_{\mathrm{J}} \uplus h_{\mathrm{F}} \\
&\qquad\qquad \wedge s, h'_{\mathrm{J}} \models J \\
&\qquad\qquad \wedge \mathrm{safe}_n(C', s', h'', J, Q))
\end{aligned}$$

- Add heap h_{J} satisfying the resource invariant, J.
- Resource invariant must be re-established in h'_{F}.

Multiple resources

$$\Gamma \models \{P\}C\{Q\} \text{ iff } \forall s\, h\, n.\ s, h \models P \implies \mathrm{safe}_n(C, s, h, \Gamma, Q)$$

$$\mathrm{safe}_0(C, s, h, \Gamma, Q) \stackrel{\text{def}}{=} \mathit{true}$$

$$\begin{aligned}
&\mathrm{safe}_{n+1}(C, s, h, \Gamma, Q) \stackrel{\text{def}}{=} \\
&\quad (C = \textbf{skip} \implies s, h \models Q) \\
&\quad \wedge (\forall h_{\mathrm{F}}.\ \neg(C, s, h \uplus h_{\mathrm{F}}) \to \textbf{abort}) \\
&\quad \wedge (\forall h_{\mathrm{J}}\, h_{\mathrm{F}}\, C'\, s'\, h'.\ (C, s \uplus h_{\mathrm{J}} \uplus h_{\mathrm{F}}, h) \to (C', s', h') \\
&\qquad\qquad \wedge s, h_{\mathrm{J}} \models \circledast_{r \in \mathit{Locked}(C') \setminus \mathit{Locked}(C)} \Gamma(r) \\
&\qquad \implies \exists h''\, h'_{\mathrm{J}}.\ h' = h'' \uplus h'_{\mathrm{J}} \uplus h_{\mathrm{F}} \\
&\qquad\qquad \wedge s, h'_{\mathrm{J}} \models \circledast_{r \in \mathit{Locked}(C) \setminus \mathit{Locked}(C')} \Gamma(r) \\
&\qquad\qquad \wedge \mathrm{safe}_n(C', s', h'', \Gamma, Q))
\end{aligned}$$

- Assume res. invariant satisfied only for acquired locks (h_{J}).
- Ensure res. invariant satisfied for released locks (h'_{J}).

Paper summary

- Yet another soundness proof for CSL
- Simple
  - Based on a standard operational semantics
  - Novel semantic definition of triples fits on a slide
- Extensible
  - Permissions
  - RGSep
  - Storable locks [Buisse, Birkedal, Støvring, MFPS 2011]
  - Concurrent abstract predicates [ECOOP 2010]
- Explains precision & conjunction rule
- Fully mechanized in Isabelle/HOL

The conjunction rule & precision

$$\frac{J \models \{P_1\}C\{Q_1\} \qquad J \models \{P_2\}C\{Q_2\} \qquad J \text{ precise}}{J \models \{P_1 \wedge P_2\}C\{Q_1 \wedge Q_2\}}$$

$$\mathrm{safe}_n(C, s, h, J, Q_1) \wedge \mathrm{safe}_n(C, s, h, J, Q_2) \implies \mathrm{safe}_n(C, s, h, J, Q_1 \wedge Q_2)$$

Recall:

$$\mathrm{safe}_{n+1}(C, s, h, J, Q) \stackrel{\mathrm{def}}{=} \ldots \wedge (\forall \ldots \implies \exists h''\, h'_{\mathrm{J}}.\ h' = h'' \uplus h'_{\mathrm{J}} \uplus h_{\mathrm{F}} \wedge s, h'_{\mathrm{J}} \models J \wedge \mathrm{safe}_n(C', s', h'', J, Q))$$

Inductive step:

1. $\exists h^1\, h^1_{\mathrm{J}}.\ h' = h^1 \uplus h^1_{\mathrm{J}} \uplus h_{\mathrm{F}} \wedge s, h^1_{\mathrm{J}} \models J \wedge \mathrm{safe}_n(C', s', h^1, J, Q)$
2. $\exists h^2\, h^2_{\mathrm{J}}.\ h' = h^2 \uplus h^2_{\mathrm{J}} \uplus h_{\mathrm{F}} \wedge s, h^2_{\mathrm{J}} \models J \wedge \mathrm{safe}_n(C', s', h^2, J, Q)$

and since J is precise, $h^1_{\mathrm{J}} = h^2_{\mathrm{J}}$, and hence $h^1 = h^2$